NASA CR. 143994

FORTRAN AUTOMATED CODE EVALUATION SYSTEM (FACES)

USER'S MANUAL

Version 2

September, 1975

(NASA-CR-143994) FORTRAN AUTOMATED CODE EVALUATION SYSTEM (FACES) USER'S MANUAL, VERSION 2 (Brown and Ramamoorthy, Inc., Berkeley, Calif.) 38 p HC $3.75 CSCL 09B

N76-10753

Unclas
G3/61 42346

BROWNE & RAMAMOORTHY, INC.

2550 Telegraph Avenue, Suite 404, Berkeley, Ca., 94704

(415) 848-0261

Table Of Contents

FORTRAN AUTOMATED CODE EVALUATION SYSTEM

USER'S MANUAL

I. FACES OBJECTIVES

The FACES system provides analysis services for FORTRAN based software systems not normally available from system software. FACES is not a compiler; compiler syntax diagnostics are not duplicated. For maximum adaptation to FORTRAN dialects, the code presented to FACES is assumed to be compiler acceptable.

The FACES system concentrates on acceptable FORTRAN code features which are likely to produce undesirable results. Emphasis is placed in the following areas:

1. Interface integrity among modules.
2. Misleading code subject to maintenance problems.
3. Keypunch errors likely to escape the compilation process.
4. Potentially malformed execution sequences.
5. Use of compiler sensitive code.

The purpose of FACES analysis is to identify potential trouble areas before they become execution time malfunctions.

While many messages indicate solid errors, messages are provided primarily to motivate user inspection of suspicious code. Messages should be evaluated by the criteria,

1. Do I understand how the mechanism works?
2. Is the feature sufficiently documented by comments or in the system documentation?

3. Would this feature confuse maintenance personnel?

4. Can a set of data values cause an error in execution?

5. Would a simple change improve code clarity?

Code features should be examined for suitability as well as "will it work".

The FACES user is assumed to be conversant with FORTRAN but at a disadvantage with code bulk and peculiarities of the FORTRAN language. FACES attempts to inform the user fully as possible what is being found as well as status information on system operation. It is the user's responsibility, however, to distinguish error potential from superficial differences in expression. Maximum advantage is gained by directing the user's intelligence to program areas most likely to cause problems.

II. OVERVIEW OF SYSTEM OPERATION

FACES is intended for application to FORTRAN based software systems composed of moderate to large number of modules. Modules usually become available for analysis in groups over a period of time. FACES is designed to incorporate new modules into the system as they become available.

In general, previously submitted modules are considered stable components. That is, once analyzed the modules usually remain unchanged.

In practice, however, occassionally need arises to modify an existing module. Rudimentary capabilities exist for replacing modules in the system; FACES is not, however, a general purpose file management system.

Anticipated operating environment. Software modules should be compiled and preferably unit tested prior to submission to the FACES system. That is, the programmer should have some level of confidence in the software.

The first set of modules presented causes a library of modules to be created. The library consists of module source code and various analysis tables which characterize the code. As new modules become available, they are analyzed and incorporated with previous software modules.

The existing modules of the software system can be analyzed by selecting inspection criteria, called queries, to be applied to the software system. Queries may be selected either while adding new source or on independent runs.

After analizing the modules for specified constructions the user can request a report. Reports are annotated program listings, truncated listing, and displays of data extracted from the software system.

The major reporting vehicle is the primary report. The primary report is an annotated module listing similar to a compiler listing with analysis messages attached to source lines where detected conditions were found.

Many problems are apparent only by considering multiple modules or portions of a single module. Secondary listings extract and display source lines from several modules or portions of a module on a single listing. For example, if a COMMON block problem is detected, considering all problems with a single block usually yields a better solution. Secondary listings minimize page flipping among various modules and hand tracing of control paths when evaluating report results.

The third report form is the display report. Display reports are organized data extracted from the software system. They differ from secondary listings in that source code is not extracted for display.

Summary of Internal Operation. FACES is roughly organized into a driver section with three subsystem components. The main driver is responsible for file manipulations and interpreting user commands. The subsystems are:

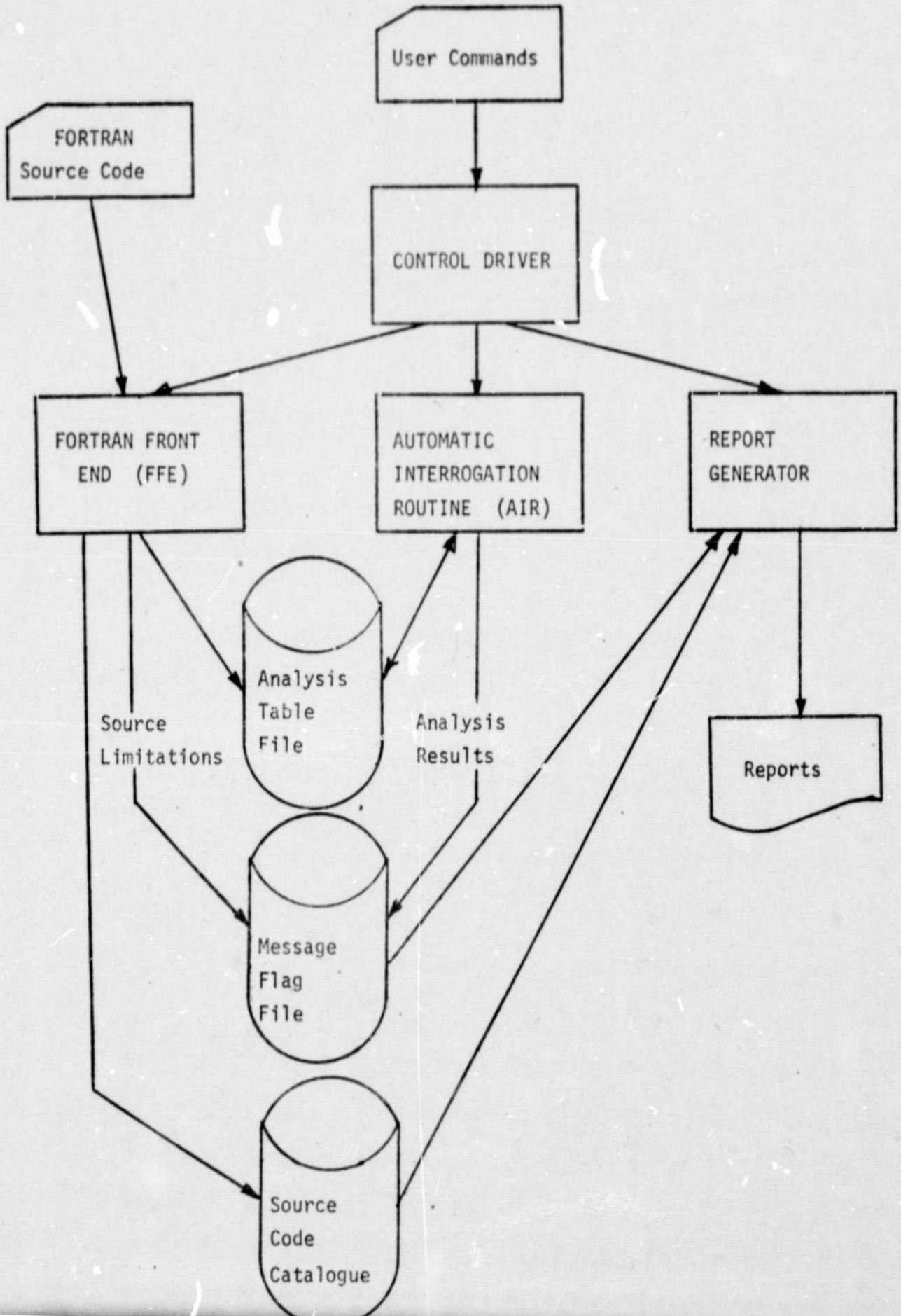
1. FORTRAN Front End (FFE). The FFE analyzes submitted source code and constructs tables which characterize module operation. New modules and references to other modules are inserted in a module Directory for system use. Source code of submitted modules is captured on a Source Code Catalogue file for use in generating reports. Situations which limit processing effectiveness are recorded on the Flag file.

2. Automatic Interrogation Routine (AIR). AIR examines the tables generated by the FFE for the constructions selected by the user. If the specified constructions are found, diagnostic messages are recorded on the Flag file.

3. Report Generator. The Report Generator combines the contents of the Flag file with the module source code to produce user reports.

Phased operation. Due to unavailable system services, the FACES system is implemented in three phases. The three phases are identical to the three sybsystems: FFE, AIR, and Report Generator. Implementation in phases restricts manipulations and reports available during a single run, but does not reduce analytic capacity.

FACES SYSTEM OVERVIEW



III. CAPABILITIES

The user controls FACES operation through command cards. Commands cause new source to be added to the software system under analysis, investigations to be selected, and results reports to be generated.

Module Directory. To catalogue the software system under analysis, a directory is maintained by FACES. Entries of the directory are all modules submitted for analysis and all references to other modules not yet defined. The directory also maintains pointers to file entries for module analysis table access and retrieval of source code.

As modules are added to the system, undefined directory entries become defined and the source code is considered in the analysis. The user controls directory entries by controlling modules submitted to the system.

Reports. The three report types are,

1. Primary listing reports. Annotated listing of modules.
2. Secondary listing reports. Truncated listing of source code participating in a query message.
3. Display reports. Displays of data extracted from the software system under analysis.

The Primary listing report contains both query analysis results and FORTRAN diagnostics issued for modules added to the system. The user controls primary reports through the REPORT command and indirectly through query selection. Default options produce reports only for modules to which messages are to be attached.

Secondary listing reports are ordinarily supplemental to the information on the primary report. Secondary reports may be suppressed through query selection. Alternatively, the secondary report can be produced in lieu of primary report messages.

Display reports are produced only if queries are selected which generate reports in the display form.

Principal user control of messages is through the query number. Each query message contains a numeric field indicating the query number which generated the message. The message may be turned off by excluding the query from the analysis list of subsequent runs.

FORTRAN Front End messages appearing on the listing cannot be controlled. They appear only when the module is analyzed for incorporation in the system.

Command Cards

Syntax. Command cards are free format single card images of 80 columns. Command items on the card are order dependent character strings terminated by a blank or a special symbol.

Blanks are separators in command card fields. Several blanks are equivalent to a single blank. Therefore, the character string AB is a single element, while A B are two separate elements.

Command items are alphanumeric symbols classified as:

1. Command key words
2. User specification controls

The general format for command cards is,

Key word ß Option list

where: Key word is an alphanumeric charater string without imbedded blanks.

ß is one or more blank columns.

Option list is a command relative list of options with specific defaults.

Available Commands. The following is a description of commands for controlling FACES operation:

INITIAL System. The INITIAL System command causes a new software system to be created for analysis. The card format is,

INITIAL SYSTEM

where SYSTEM is an optional entry in the command

The INITIAL command clears the module Directory, Source Code Catalogue, and Table File.

WARNING: The INITIAL card should be used only once to create a software system. Use of this card after the first run will cause previously created data to be destroyed.

ADD Source. The ADD source card causes new modules to be incorporated in the analysis files. The card format is,

ADD SOURCE

where SOURCE is an optional entry

The ADD command causes source code modules to be processed and added to the analysis files. New source code is added to the Source Code Catalogue. New modules participate in analysis as soon as they are added.

If a new module name is an existing Directory entry, the new module replaces the previous module. A message is issued to inform the user of these replacements. File space is not recovered on replacement.

QUERY. The QUERY command causes a software system to be examined for specific features. Queries are indentified by number (see Table I). A query list is specified by a sequence of query numbers separated by commas. For example, the construction 110,120,411, 190 specifies queries ANSIST, RESWRD, CBTYPE, UNINT. Blanks may appear between elements of the list. Numerical order is not required. The list is terminated by the absence of a comma.

QUERY commands have two forms:

Predefined QUERY group

QUERY group exceptions

where: group is an established category of queries

ALL (default) - examine all modules for all features

LOCAL - only queries which treat internal constructions of a module are selected.

GLOBAL - only queries which examine module interfaces are selected.

exceptions are an optional list of queries to be excluded from the group.

SUMMARY OF AVAILABLE QUERIES

TABLE 1

QUERY NUMBER	QUERY NAME	GROUP ALL	GROUP LOCAL	GROUP GLOBAL	Purpose
110	ANSIST	X	X		Detect misleading use of ANSI Standard function names.
120	RESWRD	X	X		Detect misleading use of FORTRAN 'Reserved' words.
130	DATVAR	X	X		Detect COMMON variables set by DATA statements outside BLOCK DATA.
140	FUNPAR	X	X		Detect FUNCTIONs which modify parameters.
150	MULBRA	X	X		Detect multiple branch statements which do not lead to next statement.
160	REDLOP	X	X		Detect redefinition of DO loop control parameters.
170	DOTERM	X	X		Detect use of DO loop control index after loop termination.
171	DOTERM	X	X		Show paths using control index after termination.
180	ASNUSE	X	X		Detect local variables assigned values but not used (possible keypunch error).
190	UNINT*				Detect uses of uninitialized local variables (possible keypunch error).
191	UNINT*				Show path of uninitialized use.
400	CBNENT	X		X	Detect COMMON blocks with different number of entries.

NUMBER	NAME	A	L	G	Purpose
401	CBNENT	X		X	Show COMMON blocks with different number of entries on a single listing.
410	CBTYPE	X		X	Detect COMMON blocks with entries having different type.
411	CBTYPE	X		X	Show COMMON blocks with type variations on a single listing.
420	CBDIM	X		X	Detect COMMON blocks with entries having different dimensions.
421	CBDIM	X		X	Show COMMON blocks with dimension variations on a single listing.
430	CBONE	X		X	Detect COMMON blocks attached to only one module (possible keypunch error on label).
440	CBNAME	X		X	Detect COMMON blocks with different names in corresponding entries.
441	CBNAME	X		X	Show COMMON blocks having different entry names on a single listing.
450	CBINDS	X		X	Detect COMMON blocks with entries having different sizes.
451	CBINDS	X		X	Show COMMON blocks with different sized entries on a single listing.
460	CBTOTS	X		X	Detect COMMON blocks of different total size.
461	CBTOTS	X		X	Show COMMON blocks of different sizes on a single listing.
500	PLNENT	X		X	Detect parameter lists having a different number of parameters.
501	PLNENT	X		X	Show calls and module definition for differing parameter lists.

NUMBER	NAME	A	L	G	Purpose
510	PLTYPE	X		X	Detect parameter lists with different types passed.
511	PLTYPE	X		X	Show CALLs and module definition for type differing parameter lists.
520	PLDIM	X		X	Detect parameter list entries with incompatable dimensions.
521	PLDIM	X		X	Show CALLs and module definition for dimension incompatable parameters.
600	CYCALL	X		X	Detect potential cyclic calling patterns among routines.

*available only through QUERY ONLY option

EXCEPT = query list

The specified queries are to be excluded from the group.

(default) - no exceptions

EXAMPLES

QUERY ALL } all queries are applied to the
QUERY } software system

QUERY LOCAL EXCEPT=120,171

Perform all local queries except RESWRD.
Do not generate path listing for DO loop index variable used after loop terminated normally.

QUERY GLOBAL EXCEPT=401,460

Perform all global queries. COMMON blocks with different number of entries will not be produced as a separate listing. COMMON blocks with different sizes will be indicated only in a separate listing.

User defined QUERY group

QUERY ONLY= query list

where query list is a user defined group of queries to be performed.

The queries specified in the query list are executed in the order specified by the user. Although numerical order is not required, maximum processing speed results from ordered query lists.

EXAMPLES

QUERY ONLY= 501, 140, 171

Causes a separate listing for parameters lists which have a different number of entries, primary listing messages for functions which modify their parameters, and paths displayed where DO loop control variables are used after normal termination.

REPORT. The REPORT command causes analysis to be displayed. The REPORT command has the form:

REPORT option

where option controls primary listing reports:

ALL - print listing of all modules in the software system being analyzed.

FLAGGED (default) - print listings for only the modules with attached messages.

Secondary reports of separate listings are not affected by the option selected.

Secondary reports are generated if selected query analysis produces secondary results. If no report is requested, a default flagged report is generated.

After a report has been processed, the analysis Flag file is set empty.

Abbreviations on commands. To minimize keypunch and spelling errors, only the first character of key words are considered in determining the command. (Note: Since the INITIAL command poses special jeopardy in the system, INITIAL command cards must contain

all alphabetic characters. Abbreviations accepted are as follows:

I - INITIAL

Q - QUERY

A - ALL

L - LOCAL

G - GLOBAL

E - EXCEPT

O - ONLY

A - ADD source

R - REPORT

A - ALL

F - FLAGGED

Command card order.

The INITIAL System command must be the first command card if a new software system is being established. For operation in phases, the following sequence is required in command cards:

PHASE 1 - INITIAL System (only for first run)
all ADD Source commands

PHASE 2 - all QUERY commands

PHASE 3 - REPORT command

If a particular command is not required on a run the command cards can be omitted. The processing phase for that command set will perform a "no-op" in this situation. For example, if no source code is to be added for an existing software system, QUERY and REPORT cards can be used to generate reports. PHASE 1 will perform no actions in this event.

Appendix A

Acceptable FORTRAN Code

Any dialect of FORTRAN may be presented to FACES although certain machine dependent forms will be excluded from the analysis. Input source code to FACES is expected to be largely ANSI Standard constructions. Commonly accepted dialectal extensions of the standard are included. If unrecognized machine dependent code is present, portions of the statement text will not participate in the analysis. A diagnostic will be issued to indicate lines of code not included in the analysis.

Input source code is assumed to be compile error free. Very limited syntax checking is performed by FACES. If a syntax error occurs which limits statement processing, the statement is truncated at the error position. In some instances, error recovery is possible; the poorly formed or unrecognized section will be skipped.

Syntax diagnostics are produced only on the run in which the source code is added to the system.

Summary of Processed Constructions. The following description is a brief review of FORTRAN constructions processed by FACES. A description of detailed constructions is presented in Appendix D.

1. Character set.

   blank character (significant only in literals)

   alphabetic characters A-Z

   numeric characters 0 - 9

   special characters = + - * / , ( ) . $ ' "

2. Card format. ANSI standard format.

3. Continuation cards. ANSI standard format.

4. Comment cards. Any character other than blank or a number in column 1 causes the card to be treated as a comment. Blank cards are comments.

5. Symbolic names. Eight character or less beginning with an alphabetic character.

6. Data types. Extended ANSI standard type definition.

Logical
Integer
Real
Double Precision
Complex
} ANSI standard

Hollerith - literal constants only
Neutral - Statement labels, Subroutine names, Common block labels, etc.
} Extended Type

7. Constants.

Integer
Real
Double Precision
} ANSI standard forms

Complex - mixed precision permitted

Logical - .TRUE. .T. .FALSE. .F.

Literal - wH char string
' char string '
" char string "

Nondecimal base constants - wZ hex chars

8. Program variables. Variable names are limited to 8 characters or less and must begin with an alphabetic character. Array dimensions are not restricted in number. Subscript references may be any arithmetic expression.

9. Operators.

   Arithmetic operators + - / * **

   Logical operators .NOT. .N. .AND. .A. .OR. .O.

   Relational operators .EQ. .NE. .GT. .GE. .LE. .LT.

10. Arithmetic expressions may contain either logical or arithmetic operators.

11. Branch targets may be either statement labels or variables set by ASSIGN statements.

Summary of Processed FORTRAN Statements. The following description is a brief summary of FORTRAN statements currently processed by FACES. A detailed description of allowable syntax is presented in Appendix D.

1. Module header card.

PROGRAM	type FUNCTION
BLOCK DATA	SUBROUTINE

2. Data declarations.

DIMENSION	IMPLICIT	EQUIVALENCE
INTEGER	COMPLEX	DOUBLE PRECISION
LOGICAL	REAL	COMMON
DATA	EXTERNAL	

3. Control statements.

   GO TO (unconditional, assigned, and computed)

   ASSIGN

   IF (3 branch arithmetic, 2 branch logical, and conditional statement execution)

   DO

   CONTINUE PAUSE STOP END

4. Assignment statements.

5. Input/Output statements.

   READ WRITE PRINT PUNCH

   END FILE REWIND BACKSPACE

6. Subprocess statements.

   ENTRY Statement Function definitions

   CALL Function References

   RETURN EXTERNAL

Excluded Statements. The following statement forms are excluded from processing.

1. FORMAT. Information from FORMAT statements is not required for current processing. The presence of FORMATs produces no diagnostic message or loss of capability.
2. NAMELIST. NAMESLIST forms vary among FORTRAN dialects. A NAMELIST statement is recognized but the variables specified are not processed. A diagnostic is issued to inform the user of omissions in the processing.

3. ENCODE/DECODE. These statement forms and operational requirements differ among FORTRAN dialects. ENCODE and DECODE statements are recognized but not processed. Variables specified in ENCODE and DECODE statements do not participate in the analysis.

Required order for FORTRAN source code. FACES is designed to minimize deck modifications in submitting code for analysis. The basic requirement is that all modules begin with a module header card (e.g. PROGRAM, SUBROUTINE, FUNCTION, and BLOCK DATA) and end with an END card. Array declarations must appear prior to the first executable use of an array element. Statement function declarations are not constrained. Internal order of other statements is not significant.

Some FORTRANs default the main program to a module not identified as a subprogram. In this case, a PROGRAM card must be inserted to identify the module in the Directory. This mechanism was selected to prevent a spurious line of source code between modules from being identified as a "new" main program, replacing valid analysis data for the actual main program.

For user convenience, Comment cards are permitted between decks. These cards are associated with the next module defined for the software system. Comment text is ignored in the analysis.

Since modules are identified by name, only one BLOCK DATA may be submitted for a software system. If multiple BLOCK DATA routines are presented, the last deck encountered is used in analysis.

Appendix B
MOTIVATION FOR QUERIES

FACES diagnostic messages are not necessarily programming errors; rather, they indicate code sections which merit inspection. The investigations should include consideration of the following aspects:

1. Vulnerability to Compiler Interpretation. ANSI Standard forms are not "perfect code", however the standard constructions define precise operations which must be implemented by compiler authors. Variant forms are selected at the compiler writer's option for implementation. If unspecified compiler features are used in the program, there is no guarantee the next version of the compiler will comply with current techniques. Compiler diagnostics are usually not generated when these changes are made.

   As a minimum, the code used for system implementation should conform to concrete operations specified in the FORTRAN user's manual. If the code will be transported to another system, only ANSI Standard forms should be employed.

   Techniques which enhance execution speed of generated code, called "optimization", may produce undesired and surprising results in execution. The programmer must be wary of certain hazardous techniques if optimization is used in the compiler.

2. Potential Misinterpretation. Program operation is almost always clear to the author in the beginning. To a second

party, style and uniformity are as important to comprehension as algorithm completeness. Hidden variations in form are invitations to future failure during maintenance and modification.

3. Implementation Errors. Even thoroughly tested code is subject to implementation errors. Often, the "right" test case has not yet been encountered. Keypunch and coding errors frequently produce valid FORTRAN code. These problems are more quickly surfaced if "unusual" coding patterns are displayed for user inspection.
4. Interface Fumbles. Interface consistency reduces chances of incorrect operation. Since module interface checks are difficult to verify manually, unusual interface operations are indicated by FACES analysis.
5. Runtime Linkage. Operations among modules in the normal compile/link/load mode are not precisely defined by the ANSI Standard. Many surprises await large software systems with imperfect calling protocalls. Little diagnostic information on module linkage error is provided by computer support software.

Path Tracing. FACES does not consider program logic in path tracing, hence "impossible" paths may be produced. When considering these paths, the user should be aware that "impossible" conditions may become "possible" in error environments or in the expanded scope of future modifications. The cost of a few additional assignment statements

or extra tests must be carefully weighed against the cost of potential future malfunctions.

Individual Query Considerations. The following discussion is intended to assist the user in the evaluation of malfunction potential as indicated by messages.

110 - ANSIST - The use of an ANSI Standard function name as a program variable may mislead maintenance personnel. The potential for misinterpretation is especially high if the variable is subscripted.

The use of an ANSI Standard function name as a software module name is both misleading and dangerous. If another module attempts to use the standard function, linkage will normally be generated to this software module. Thus, the use of this name will exclude the use of the standard function in the system.

Defining a function with the same name as an ANSI Standard name is even more problematic. In some compilers, many standard functions are effectively compiled "in-line" -- no linkage is created to external modules. As a result, the code will evaluate the standard function rather than call the user's routine.

If optimization is performed by the compiler, the user's routine reference may be mistaken for a reference to the standard function. This may result in the optimization creating incorrect code.

120 - RESWRD - Use of FORTRAN "reserved" words makes code difficult to read. How would you like to follow code like:

DO 1.' = IT(K) (an assignment statement).

130 - DATVAR - Data statements to COMMON variables in routines other than BLOCK DATA are contrary to the ANSI Standard. If this restriction is intended to protect the user from initializing the same variable more than once. Multiple initializations make program operation "load dependent" (i.e., the initial value is set by the last module loaded).

If the system is required to operate in an overlay fashion, DATA statements may reinitialize the COMMON variable on each overlay; alternatively, the variable may not be re-initialized each time. Exact operation is machine dependent. BLOCK DATA, however, is usually loaded once.

140 - FUNPAR - Functions which modify their parameters can lead to unexpected results. Since the output from a function is through the function name, programmers are less cautious of parameter side effects. Input parameters to functions are frequently constants; modification of a constant may upset the calling routine. Since the error occurs on the other side of an interface, this error is particularly difficult to find.

For example,

```
      Y = FUNC(2., X*3)
          .
          .
          .
      FUNCTION  FUNC(A, B)
      A = A**2     (2. is now the value 4.)
```

Who would expect 2. to suddenly change value?

Another problem with side effect functions occurs if compiler optimization is performed. If an expression, for example X*3, is changed, a bad value may be passed to a later computation containing the "common subexpression" X*3.

Since functions may appear in IF statement conditions, compilers may not execute the function as often as the user expects. Some logical conditions are compiled to branch as soon as a .TRUE. or .FALSE. value is determined; if the function appears several times in the logical expression, the function may only be referenced once. For example, consider,

```
IF( FUNC(I) .OR. FUNC(J) ) B = C
```

If the first FUNC reference produces .TRUE. , the second reference may not occur at all.

150 - MULBRA - Multiple branching statements which do not branch to the statement immediately following, are necessarily errors. There is a strong chance the branch list is incorrect. If correct, this type of branch is rather unusual and may represent a "special case" or code "patch" to be removed in the future.

160 - REDLOP - The redefinition of DO loop control variables within the loop body violates the ANSI Standard. Loop operation is usually unpredictable. At the very least, this technique is highly compiler sensitive.

170 - DOTERM - Use of the DO loop index variable after normal loop termination is "undefined" by the ANSI Standard. The value assigned the variable can be quite surprising, depending upon compiler implementation. Using an assignment statement at the end of the loop will normalize operation.

If this use is detected, a secondary listing will be produced showing paths on which the index variable is used. (See "Flow of Control Path Tracing".)

180 - ASNUSE - Local variables assigned values but never used are often keypunch errors or historical legacies. They may be caused by errors in program logic.

190 & 191 - UNINT - The use of an uninitialized local variable is caused by either keypunch errors or errors in program logic. The paths taken to the uninitialized use are available on a secondary report. (See "Flow of Control Path Tracing".)

400 - 461 - COMMON Block Misalignment

COMMON Blocks which are not identical are usually "errors waiting to happen". Maintenance is much less error-prone where uniform COMMON structures are employed. The efficiency of mixing modes and changing structures must be carefully weighed against difficulty in interpretation.

Keypunch errors are particularly troublesome in COMMON. A mispunch can create a new labeled BLOCK totally detached from the rest of the system. A misspelled or transposed variable name may not be currently used; if referenced during modification, the programmer can expect a long and tedious search for the error.

500 - 521 - Parameter List Misalignment

Parameter alignment problems frequently lead to execution errors. They may be caused by keypunch errors or imprecise interface definition.

Parameter count and type problems are obvious. Incompatible argument dimensions are either outright errors or invitations to malfunction. The use of differing dimensions should be approached with caution.

600 - CYCALL - Cyclic calling sequences usually result in infinite loops unless recursive calling is explicitly supported. Upon closing the cycle, the original entry return address is destroyed. This makes it impossible to "get back" to the initial routine.

Appendix C

Deck Set-Up

It is beyond the scope of this manual to present a detailed discussion of how to use JCL to set up the execution of FACES. This task is left to the user, although examples are presented.

A number of files must be allocated by the user. Some are direct (random) access files; most are sequential access files. Some files are needed in only one phase; other files are needed in all phases of FACES execution.

FACES uses eight files, which are listed below. Unless otherwise noted, each is a sequential file.

2 - FLAG: Flag File (FMSG: Fortran Message File)

3* - TABL: Table File

4* - SCAT: Source Code Catalogue File

6 - PRNT: Print File (Output File)

8 - ANSI: ANSI Standards Function Names File

9 - CTRL: Control File

10 - SCIN: Source Code Input File

11 - RESW: Fortran Reserved Word File

*direct (random) access file

These files are used during the following execution phases:

Phase 1	Phase 2	Phase 3
2 - FLAG	2 - FLAG	2 - FLAG
3 - TABL	3 - TABL	3 - TABL
4 - SCAT	6 - PRNT	4 - SCAT
6 - PRNT	8 - ANSI	6 - PRNT
9 - CTRL	9 - CTRL	9 - CTRL
10 - SCIN	11 - RESW	

Significant File Sizes:

FLAG - 76 bytes, 10,000 records

TABL - 400 bytes, 24,000 records

SCAT - 80 bytes, 100,000 records

FACES normally operates in four steps:

1) Phase 1 (Fortran Front End)
2) Phase 2 (Automatic Interrogation Routine)
3) Sorting of FLAG File
4) Phase 3 (Report Generator)

Before Phase 1 can execute, the source code to be analyzed must reside in the SCIN File, and the command cards used in Phase 1 must reside in the CTRL File. Allowable Command Cards for this phase are

INITIAL SYSTEM and

ADD SOURCE

During Phase 1 execution, flags are placed in the FLAG File, tables are produced and placed in the TABL File, the source code is catalogued and placed in the SCAT File, and all messages to be printed pass through the PRNT File.

If a new software system is to be analyzed, the CTRL File must consist of

INITIAL SYSTEM and

ADD SOURCE

If new modules are to be added to the software system being analyzed, the CTRL File must consist of

ADD SOURCE

and the TABL and SCAT files must contain the information created on a previous run which depicts the software system being analyzed. (The TABL and SCAT files must be saved after a run if future analysis is desired which bypasses Phase 1.)

The SCIN File may be discarded after Phase 1.

Phase 2 uses two files used nowhere else in FACES. These files are ANSI and RESW. They are "read only" files. They are not needed if the user does not invoke queries 110 and 120 (search for ANSI Standards Function Names or Fortran Reserved Words used as user-defined names).

The CTRL File may contain only

QUERY _____

commands.

During Phase 2 execution, flags are placed in the FLAG File, tables are read in from the TABL File or are produced and placed in the TABL File, all messages to be printed pass through the PRNT File, and ANSI and RESW are used as "read only" files for certain queries.

The sorting step involves only the FLAG File. Each record in the file has the following format:

FORMAT (5(2X,I5), 2X, 2A4, 3(2X,I5), 2X, 2A4)

The sort is performed on the first seven integer fields, left-to-right, in ascending order. The sort is performed through JCL; FACES has no sorting capabilities. For further discussions, see the detailed description of the FLAG File.

For Phase 3, the CTRL File consists of the command,

REPORT _____

This controls what information is to be printed during Phase 3.

During Phase 3 execution, flags are read in from the sorted FLAG File, information is gleamed from the TABL and SCAT files, and everything to be printed passes through the PRNT File.

Examples:

To analyze a software system completely for the first time, a deck might appear as

```
JCL
 .
 .
 .
INITIAL SYSTEM
ADD SOURCE
JCL
 .
 .
 .
QUERY ALL
QUERY ONLY = 190,191
JCL
 .
 .
 .    (Sort)
REPORT FLAGGED
JCL
 .
 .
 .
```

To add modules to a software system which has already been analyzed and whose TABL and SCAT Files have been saved, and needing only local analysis, a deck might appear as

```
JCL
 .
 .
 .
ADD SOURCE
JCL
 .
 .
 .
QUERY LOCAL
JCL
 .
 .
 .    (Sort)
REPORT FLAGGED
JCL
 .
 .
 .
```

To perform a global analysis of a software system which already has been analyzed and whose TABL and SCAT Files have been saved, and needing a listing of all source code analyzed, a deck might appear as

```
JCL
 .
 .
 .
QUERY GLOBAL
JCL
 .
 .
 .    (Sort)
REPORT ALL
JCL
 .
 .
 .
```

Note that Phase 1 has been bypassed.

```
//NC011449 JOB (,                                                    JOB 35
// A09,81),1HNISV2E6140CAPPS,NSGLEVEL=1
//GOPH1 EXEC PGM=PHASE1,TIME=3,REGION=300K
//STEPLIB DD DSN=BANDR,UNIT=DISK,DISP=(SHR,KEEP),VOL=SER=CSC001
//FT02F001 DD DSN=EEFLAG,UNIT=DISK,DISP=(NEW,PASS),
//    SPACE=(76,(100,20)),DCB=(LRECL=76,RECFM=FB,BLKSIZE=76)
//FT03F001    DD DSN=EETABL,UNIT=DISK,DISP=(NEW,PASS),
//    SPACE=(400,(2400)),DCB=(RECFM=F,LRECL=400,BLKSIZE=400)
//FT04F001    DD DSN=EESCAT,UNIT=DISK,DISP=(NEW,PASS),
//    SPACE=(80,(1000)),DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//FT06F001    DD SYSOUT=A
//FT09F001    DD *            CTRL FILE
//FT10F001    DD *            SCIN FILE
//SYSUDUMP    DD SYSOUT=A
IEF236I ALLOC. FOR NC011449 GOPH1
IEF237I 154   ALLOCATED TO STEPLIB
IEF237I 132   ALLOCATED TO FT02F001
IEF237I 132   ALLOCATED TO FT03F001
IEF237I 132   ALLOCATED TO FT04F001
IEF237I 473   ALLOCATED TO FT06F001
IEF237I 433   ALLOCATED TO FT09F001
IEF237I 434   ALLOCATED TO FT10F001
IEF237I 474   ALLOCATED TO SYSUDUMP
IEF142I - STEP WAS EXECUTED - COND CODE 0000
IEF285I   BANDR                                          KEPT
IEF285I   VOL SER NOS= CSC001.
IEF285I   SYS75230.T112604.RV000.NC011449.FLAG           PASSED
IEF285I   VOL SER NOS= 222222.
IEF285I   SYS75230.T112604.RV000.NC011449.TABL           PASSED
IEF285I   VOL SER NOS= 222222.
IEF285I   SYS75230.T112604.RV000.NC011449.SCAT           PASSED
IEF285I   VOL SER NOS= 222222.
IEF373I STEP /GOPH1   / START 75230.1126
IEF374I STEP /GOPH1   / STOP  75230.1218 CPU   2MIN 21.32SEC MAIN 228K LCS   0K
//GOPH2 EXEC PGM=PHASE2,TIME=5,REGION=300K
//STEPLIB DD DSN=BANDR,UNIT=DISK,DISP=(SHR,KEEP),VOL=SER=CSC001
//FT02F001 DD DSN=EEFLAG,DISP=(OLD,PASS)
//FT03F001 DD DSN=EETABL,DISP=(OLD,PASS)
//FT05F001 DD SYSOUT=A
//FT08F001 DD DSN=ANSI,UNIT=DISK,DISP=(SHR,KEEP),VOL=SER=CSC001
//FT09F001 DD *               CTRL FILE
//FT11F001 DD DSN=RESW,UNIT=DISK,DISP=(SHR,KEEP),VOL=SER=CSC001
//SYSUDUMP DD SYSOUT=A
IEF236I ALLOC. FOR NC011449 GOPH2
IEF237I 154   ALLOCATED TO STEPLIB
IEF237I 132   ALLOCATED TO FT02F001
IEF237I 132   ALLOCATED TO FT03F001
IEF237I 473   ALLOCATED TO FT06F001
IEF237I 154   ALLOCATED TO FT08F001
IEF237I 435   ALLOCATED TO FT09F001
IEF237I 154   ALLOCATED TO FT11F001
IEF237I 474   ALLOCATED TO SYSUDUMP
IEF142I - STEP WAS EXECUTED - COND CODE 0000
IEF285I   BANDR                                          KEPT
IEF285I   VOL SER NOS= CSC001.
IEF285I   SYS75230.T112604.RV000.NC011449.FLAG           PASSED
IEF285I   VOL SER NOS= 222222.
IEF285I   SYS75230.T112604.RV000.NC011449.TABL           PASSED
IEF285I   VOL SER NOS= 222222.
IEF285I   ANSI                                           KEPT
IEF285I   VOL SER NOS= CSC001.

IEF285I   RESW                                           KEPT
IEF285I   VOL SER NOS= CSC001.
IEF373I STEP /GOPH2   / START 75230.1218
IEF374I STEP /GOPH2   / STOP  75230.1222 CPU   0MIN 43.49SEC MAIN 254K LCS   0K
//TSORT EXEC SORTD
XXSORT    EXEC PGM=IERRCOOO,REGION=26K                                  20000013
XXSYSOUT  DD   SYSOUT=A                                                 40000013
XXSORTLIB DD   DSNAME=SYS1.SORTLIB,DISP=SHR                             60000013
//SORT.SORTIN  DD DSN=EEFLAG,DISP=(OLD,DELETE)
//SORT.SORTOUT DD DSN=EEFLG1,DISP=(NEW,PASS),UNIT=DISK,
//    SPACE=(76,(100,20),RLSE),
//    DCB=(RECFM=FB,LRECL=76,BLKSIZE=76)
```

```
IEF285I   RESW                                          KEPT
IEF285I   VOL SER NOS= CSC001.
IEF373I STEP /GOPH2   / START 75230.1218
IEF374I STEP /GOPH2   / STOP  75230.1222 CPU   0MIN 43.49SEC MAIN 254K LCS   0K
//TSORT EXEC SORTD
XXSORT   EXEC PGM=IERRCOOO,REGION=26K                                    20000013
XXSYSOUT DD   SYSOUT=A                                                   40000013
XXSORTLIB DD  DSNAME=SYS1.SORTLIB,DISP=SHR                               60000013
//SORT.SORTIN   DD DSN=EEFLAG,DISP=(OLD,DELETE)
//SORT.SORTOUT DD DSN=EEFLG1,DISP=(NEW,PASS),UNIT=DISK,
//    SPACE=(76,(100,20),RLSE),
//    DCB=(RECFM=FB,LRECL=76,BLKSIZE=76)
//SORT.SORTWK01 DD UNIT=DISK,SPACE=(CYL,(1))
//SORT.SORTWK02 DD UNIT=DISK,SPACE=(CYL,(1))
//SORT.SORTWK03 DD UNIT=DISK,SPACE=(CYL,(1))
//SORT.SORTWK04 DD UNIT=DISK,SPACE=(CYL,(1))
//SORT.SYSIN DD *
IEF236I ALLOC. FOR N0011449 SORT      TSORT
IEF237I 473   ALLOCATED TO SYSOUT
IEF237I 151   ALLOCATED TO SORTLIB
IEF237I 132   ALLOCATED TO SORTIN
IEF237I 132   ALLOCATED TO SORTOUT
IEF237I 132   ALLOCATED TO SORTWK01
IEF237I 132   ALLOCATED TO SORTWK02
IEF237I 132   ALLOCATED TO SORTWK03
IEF237I 132   ALLOCATED TO SORTWK04
IEF237I 436   ALLOCATED TO SYSIN
IEF142I - STEP WAS EXECUTED - COND CODE 0000
IEF285I   SYS1.SORTLIB                                  KEPT
IEF285I   VOL SER NOS= NASAG1.
IEF285I   SYS75230.T112604.RV000.N0011449.FLAG          DELETED
IEF285I   VOL SER NOS= 222222.
IEF285I   SYS75230.T112604.RV000.N0011449.FLG1          PASSED
IEF285I   VOL SER NOS= 222222.
IEF285I   SYS75230.T112604.RV000.N0011449.R0000009      DELETED
IEF285I   VOL SER NOS= 222222.
IEF285I   SYS75230.T112604.RV000.N0011449.R0000010      DELETED
IEF285I   VOL SER NOS= 222222.
IEF285I   SYS75230.T112604.RV000.N0011449.R0000011      DELETED
IEF285I   VOL SER NOS= 222222.
IEF285I   SYS75230.T112604.RV000.N0011449.R0000012      DELETED
IEF285I   VOL SER NOS= 222222.
IEF373I STEP /SORT    / START 75230.1222
IEF374I STEP /SORT    / STOP  75230.1223 CPU   0MIN 03.37SEC MAIN  60K LCS   0K
//GOPH3 EXEC PGM=PHASE3,TIME=3,REGION=100K
//STEPLIB DD DSN=BANDR,UNIT=DISK,DISP=(SHR,KEEP),VOL=SER=CSC001
//FT02F001     DD DSN=EEFLG1,DISP=(OLD,DELETE),UNIT=DISK
//FT03F001     DD DSN=EETABL,DISP=(OLD,PASS)
//FT04F001     DD DSN=EESCAT,DISP=(OLD,DELETE)
//FT06F001 DD SYSOUT=A
//FT09F001 DD *                  CTRL FILE
//SYSUDUMP DD SYSOUT=A
//
IEF236I ALLOC. FOR N0011449 GOPH3
IEF237I 154   ALLOCATED TO STEPLIB
IEF237I 132   ALLOCATED TO FT02F001
IEF237I 132   ALLOCATED TO FT03F001
IEF237I 132   ALLOCATED TO FT04F001
IEF237I 473   ALLOCATED TO FT06F001
IEF237I 437   ALLOCATED TO FT09F001
IEF237I 475   ALLOCATED TO SYSUDUMP


IEF142I - STEP WAS EXECUTED - COND CODE 0000
IEF285I   BANDR                                         KEPT
IEF285I   VOL SER NOS= CSC001.
IEF285I   SYS75230.T112604.RV000.N0011449.FLG1          DELETED
IEF285I   VOL SER NOS= 222222.
IEF285I   SYS75230.T112604.RV000.N0011449.TABL          PASSED
IEF285I   VOL SER NOS= 222222.
IEF285I   SYS75230.T112604.RV000.N0011449.SCAT          DELETED
IEF285I   VOL SER NOS= 222222.
IEF373I STEP /GOPH3   / START 75230.1223
IEF374I STEP /GOPH3   / STOP  75230.1225 CPU   0MIN 08.02SEC MAIN  74K LCS   0K
```

```
IEF237I 437   ALLOCATED TO SYS00001
IEF237I 475   ALLOCATED TO SYSUDUMP


 IEF142I - STEP WAS EXECUTED - COND CODE 0000
 IEF285I   BANOR                                          KEPT
 IEF285I   VOL SER NOS= CSC0C1.
 IEF285I   SYS75230.T112604.RV000.N0011449.FLG1           DELETED
 IEF285I   VOL SER NOS= 222222.
 IEF285I   SYS75230.T112604.RV000.N0011449.TABL           PASSED
 IEF285I   VOL SER NOS= 222222.
 IEF285I   SYS75230.T112604.RV000.N0011449.SCAT           DELETED
 IEF285I   VOL SER NOS= 222222.
 IEF373I STEP /GOPH3    / START 75230.1223
 IEF374I STEP /GOPH3    / STOP  75230.1225 CPU    0MIN 08.02SEC MAIN 7[illegible]  OK
 IEF285I   SYS75230.T112604.RV000.N0011449.TABL           DELETED
 IEF285I   VOL SER NOS= 222222.
 IEF375I  JOB /N0011449/ START 75230.1126
 IEF376I  JOB /N0011449/ STOP  75230.1225 CPU   3MIN 16.20SEC
```

ORIGINAL PAGE IS
OF POOR QUALITY